UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anže Hribar

Krmiljenje ogrevanja z vgrajenim sistemom

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2017

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anže Hribar

Krmiljenje ogrevanja z vgrajenim sistemom

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Robert Rozman

Ljubljana, 2017

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Izdelajte zasnovo sistema za učinkovito krmiljenje starejših ogrevalnih sistemov, ki jih nadgradimo z novimi viri toplote. Sistem tudi realizirajte z vso potrebno strojno in programsko opremo. Proces ogrevanja poskusite optimizirati, predvsem z vidika stroškov porabljene energije. Hkrati omogočite uporabniku vpogled v delovanje sistema in možnost vpliva na glavne nastavitve sistema. Ustrezno poskrbite tudi za možnost odpovedi delovanja posameznih delov sistema.

Hvala mentorju dr. Robertu Rozmanu za vso potrpežljivost in pripravljenost za pomoč. Hvala moji družini, ženi Neži in otrokom Avguštinu, Klemnu in Petru. Hvala mojim staršem, da so me vedno podpirali pri študiju. Hvala sestricama. Hvala Bogu.

Moji družini.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
FSM	Finite-State Machine	Končni avtomat stanj
LCD	Liquid-crystal display	Zaslon s tekočimi kristali
NVM	Non-volatile memory	Brezizgubni pomnilnik
GPIO	General purpose input/output pin	Splošnonamenski vhodno izhodni priključek
RAM	Random-access memory	Pomnilnik z naključnim dostopom
EEPROM	Electrically erasable programmable read-only memory	Električno izbrisljivi programirljivi bralni pomnilnik
CPE	Central processing unit	Centralno procesna enota
AC	Alternating current	Izmenični tok
UTP	Unshielded twisted pair	Parica
PSP	Interrupt service routine (ISR)	Prekinitveno servisni program
CNC	Computer numeric control	Računalniško numerično krmiljenje
IDE	Integrated development environment	Razvojno okolje
CAD	Computer-aided design	Računalniško podprto načrtovanje
CAM	Computer-aided manufacturing	Računalniško podprta proizvodnja
LED	Light-emitting diode	Svetleča dioda

Povzetek

Naslov: Krmiljenje ogrevanja z vgrajenim sistemom

V diplomski nalogi se načrtuje in implementira prototip naprave, ki rešuje problem poganjanja obtočnih črpalk v starejših ogrevalnih sistemih, ki so običajno brez obstoječe krmilne elektronike. Elektronsko krmiljenje se potrebuje tam, kjer mora več različnih sklopov – toplotna črpalka in centralna peč – tvoriti usklajeno celoto. Realizira se enostavna in poceni rešitev, ki bo modularna in jo je kasneje mogoče nadgraditi. S pomočjo končnega avtomata stanj se razvije uporabniški vmesnik, ki omogoča, da uporabnik poleg spremljanja vseh parametrov lahko tudi spreminja nastavitve naprave. Ker je potrebno, da se nastavitve ob morebitnem izklopu napajanja obdržijo, so razvite funkcije za branje in pisanje podatkov v bliskovni pomnilnik. Naredi se tiskano vezje, ki je zasnovano modularno, da se kasneje lahko doda temperaturna tipala ali rele kartica, za krmiljenje dodatnih elementov. Predvidi in opiše se tudi ravnanje v primeru napak – varni način delovanja. S pomočjo programske opreme CAD/CAM se načrtuje ohišje, ki se ga izdela s pomočjo rezkalnika CNC. Končni prototip se vgradi v ogrevalni sistem, kjer se tudi testira in preizkusi delovanje.

Ključne besede: ogrevalni sistem, STM32F4 Discovery, uporabniški vmesnik, končni avtomat stanj

Abstract

Title: Embedded System for Heating Control

The present thesis proposes a design and implementation of a functional embedded system prototype, which solves a problem of multiple circulator pumps in old heating systems without any kind of control electronics. Control electronics is needed where multiple units of a heating system - namely the heat pump and the central heating unit - must function as one unit. The thesis develops an easy, affordable and modular solution with a possibility of upgrading. User interface is implemented using a finite-state machine, which enables user to change settings. Since user settings must retain their value in case of power shortage, functions of reading from flash memory and writing to flash memory have been developed. Printed circuit board (PCB) is modular, so that new sensors and actuators can be added in the future. In case of errors, a special error-handling mode is initiated - a so-called safe mode. A front plastic housing mask is designed with the help of CAD/CAM software, to be milled with CNC. Finally, the prototype is installed and tested on a heating system.

Keywords: heating system, STM32F4 Discovery, user interface, finite-state machine

Poglavje 1 Uvod

V zadnjih desetletjih je na področju ogrevanja stanovanjskih objektov prišlo do velikih sprememb. Lastniki starejših hiš prepoznavajo potrebo po prenovi ogrevalnega sistema, vendar takšne odločitve pogosto zahtevajo menjavo celotnega centralnega ogrevalnega sistema. Kadar lastniki želijo svoj starejši ogrevalni sistem delno posodobiti, lahko pride do problema, kako takšen sistem ogrevanja krmiliti. Sodobnih virov toplote se ne krmili mehansko, temveč elektronsko, zato je potrebno v ogrevalni sistem namestiti elektronsko krmilje. Na trgu sicer obstajajo že izdelane rešitve, vendar njihova cena ne opravičuje krmiljenja enega samega podsklopa ogrevalnega sistema, zato nam ostaneta samo še dve možnosti: ročni nadzor ali samogradnja.

Pričujoče diplomsko delo rešuje problem krmiljenja dveh obtočnih črpalk v ogrevalnih sistemih, ki nimajo vgrajene krmilne elektronike. Poiskati želimo enostavno in poceni rešitev, ki bo modularna in jo bo kasneje mogoče po potrebi nadgraditi.

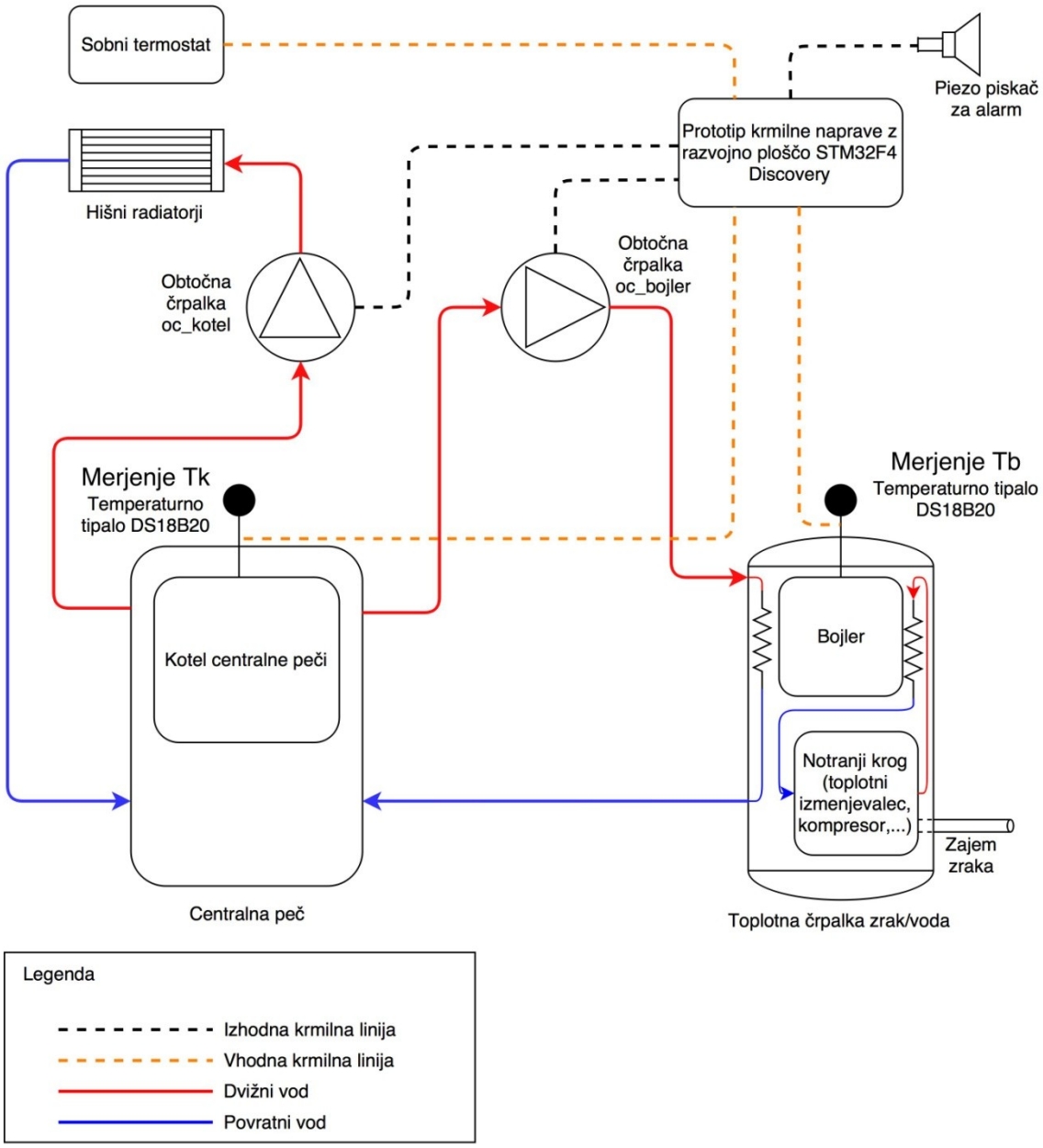
Poglavje 2 Zasnova in delovanje sistema

Razvili smo napravo, ki rešuje problem krmiljenja obtočnih črpalk v ogrevalnih sistemih, ki nimajo krmilne elektronike. Elektronsko upravljanje potrebujemo tam, kjer mora več različnih sklopov, kot so npr. gorilec za peč, toplotna črpalka, solarni paneli ipd., tvoriti usklajeno celoto. V našem primeru gre za centralno peč na olje in toplotno črpalko.

Centralna peč segreva vodo za ogrevanje hiše, vodo v grelniku (bojlerju) toplotne črpalke pa uporabljamo za sanitarno vodo.

Toplotna črpalka segreva sanitarno vodo do določene navzgor omejene temperature. Ker se v njej zadržuje topla voda, obstaja nevarnost, da se v bojlerju namnoži človeku nevarna bakterija legionela. Najugodnejše okolje za razvoj te bakterije je voda s temperaturo med 40 in 55 °C, zato moramo vodo v bojlerju toplotne črpalke segreti vsaj na 60 °C [1].

Toplotna črpalka je tipa zrak/voda, kar pomeni, da se za segrevanje vode uporabi zrak iz okolice. Kadar so pogoji za delovanje toplotne črpalke neugodni (zelo hladen zrak) je potrebno sanitarno vodo v bojlerju toplotne črpalke segreti z drugim virom toplote, v našem primeru s centralno pečjo. Med centralno pečjo in toplotno črpalko zato poteka vod, na katerega je nameščena obtočna črpalka, ki poganja vročo vodo iz centralne peči v bojler toplotne črpalke (Slika 2.1). Ta obtočna črpalka je torej prva, ki jo moramo krmiliti. Elektronika požene obtočno črpalko le v primeru, ko je voda v bojlerju toplotne črpalke hladna, oz. je pod neko mejno vrednostjo, in ko je temperatura vode v centralni peči večja kot temperatura vode v bojlerju toplotne črpalke.



Slika 2.1: Shema ogrevalnega sistema s prototipom krmilne naprave.

Kot drugo krmilimo obtočno črpalko centralnega ogrevanja. V hiši je nameščen sobni termostat. Ko temperatura v hiši pade pod nastavljeno vrednost na termostatu, ta sklene stikalo in vklopi obtočno črpalko centralne peči. Ker običajno ogrevalni sistem nima krmilne elektronike, se pojavi težava. Sobni termostat vklaplja samo obtočno črpalko, nima pa podatka, ali je voda v kotlu centralne peči sploh dovolj vroča. Ko termostat vklopi črpalko, je voda v centralni peči lahko hladna in po sistemu kroži hladna voda. Naše krmilje mora torej

preverjati oboje: temperaturo vode v kotlu centralne peči in stanje hišnega termostata. Če sta oba pogoja izpolnjena, potem vklopi obtočno črpalko centralnega ogrevanja.

Pri iskanju optimalne rešitve za opisani problem se osredotočamo na naslednje cilje:

- možnost upravljanja osnovnih funkcij prek zaslona LCD,
- enostavnost za uporabnika,
- cenovna dostopnost,
- možnost kasnejših nadgradenj, modularnost.

Poglavje 3 Strojna oprema

Izdelava naprave zahteva izbiro strojne opreme, ki bo ustrezala zastavljenim ciljem. Za krmiljenje sistema potrebujemo krmilno enoto, mikrokrmilnik z ustreznimi vhodno izhodnimi napravami, temperaturna tipala, zaslon LCD s tipkami, močnostne krmilne elemente, kot so releji ali polprevodniški releji in zvočnik ali piezo piskač.

3.1 Potrebne funkcionalnosti krmilne enote

Izbiramo med razvojnimi ploščami Arduino, Raspberry Pi in STM32F4 Discovery. Glede na zastavljene cilje izberemo krmilno enoto, ki bo ustrezala naslednjim zahtevam:

- veliko število priključkov GPIO,
- zmogljiv mikrokrmilnik,
- vgrajen pomnilnik NVM za shranjevanje spremenljivk ob izklopu napajanja,
- vsaj trije časovniki,
- ustrezno razvojno okolje in
- razhroščevalnik.

3.1.1 Priključki GPIO

Ker želimo, da bo elektronika v ogrevalnem sistemu nadgradljiva, poleg krmiljenja modula LCD, temperaturnih tipal, tipk, relejev in alarma potrebujemo veliko priključkov GPIO, da nam ob dodajanju funkcionalnosti slednjih ne zmanjka.

3.1.2 Zmogljiv mikrokrmilnik

Danes so mikrokrmilniki že tako zmogljivi, da bi bila skoraj vsaka izbira primerna. Kljub temu izberemo nekoliko zmogljivejšega, ki bo ustrezal tudi naknadnim razširitvam, kot so upravljanje ogrevalnega sistema prek spleta, obveščanje uporabnika prek storitve kratkih sporočil (angl. SMS), upravljanje s pametnim telefonom in dodajanje različnih tipal.

3.1.3 Pomnilnik NVM

V napravi za krmiljenje ogrevalnega sistema bo nekatere nastavitve možno spreminjati glede na potrebe uporabnika. Te nastavitve se ob izklopu napajanja ne smejo izgubiti oziroma ponastaviti, saj bi moral v nasprotnem primeru uporabnik želene nastavitve vedno znova vnašati. V ta namen napravi vgradimo pomnilnik NVM. Spremenljivke v delovnem pomnilniku RAM namreč izgubijo vrednost takoj po izklopu napajanja, kar se pri pomnilnikih NVM ne zgodi.

Izbiramo lahko med bliskovnimi pomnilniki (angl. flash) in pomnilniki EEPROM. Slednji tipično ponujajo 10-krat več pisalnih ciklov kot bliskovni pomnilnik (100.000 proti 10.000). V naši aplikaciji ne pričakujemo tako velikega števila ciklov, saj bo uporabnik parametre spreminjal samo na začetku. Ko bo sistem nastavljen, se parametri ne bodo več pogosto spreminjali, zato bliskovni pomnilnik popolnoma zadošča.

3.1.4 Časovniki

Pri vgrajenih sistemih igrajo pomembno vlogo časovniki (angl. timer). Brez teh bi bila koda programa nepregledna in težko berljiva. Naš projekt zahteva tri časovnike. Sistemski časovnik (angl. system timer) vsakih nekaj milisekund prek prekinitev požene glavno zanko krmilnega programa. Drugi časovnik meri čas, odkar smo zadnjič pritisnili tipko, tretjega pa uporabimo za prekinjeno piskanje alarma ob napaki. Ob dodajanju funkcionalnosti se lahko pokaže potreba po dodatnih časovnikih, zato je število le-teh eden izmed pomembnih kriterijev pri izbiri krmilne enote.

3.1.5 Razhroščevalnik

Razhroščevalnik (angl. debugger) je programsko orodje, ki ga v kombinaciji s strojnim vmesnikom uporabljamo za preizkušanje in razhroščevanje drugih programov. Razhroščevanje pomeni odpravljanje napak. Ko pride do izpada programa, razhroščevalnik pokaže položaj v originalni izvorni kodi, v nekaterih primerih pa kar v strojni kodi prevedenega programa, ki ga preizkušamo.

Program izpade oz. se nepričakovano ustavi zaradi programske napake. Morda na danem mestu CPE poskuša izvajati dostop do nedostopne lokacije v pomnilniku [2].

Razvijanje programa si je torej težko predstavljati brez razhroščevalnika, saj z njegovo pomočjo lahko ustavimo urin signal procesorju na mikrokrmilniku, korak za korakom izvajamo ukaze in spremljamo vrednosti posameznih spremenljivk. To nam olajša iskanje napak v našem programu ter pohitri razvoj.

3.2 Izbira krmilne enote

Glede na zgoraj opisane zahteve smo v ožji izbor uvrstili 3 razvojne plošče: STM32F4 Discovery [3], Raspberry Pi 2 [4] in Arduino UNO [5]. Vsaka ima svoje prednosti in slabosti (Tabela 3.1).

	Prednosti	**Slabosti**
STM32F4 Discovery	⊕ Zmogljiv procesor, ⊕ veliko število priključkov GPIO, ⊕ razhroščevalnik, ⊕ velik bliskovni pomnilnik, ⊕ 14 časovnikov.	⊖ Nima vgrajenega EEPROM pomnilnika.
Raspberry Pi 2	⊕ Zelo zmogljiv procesor, ⊕ zadostno število priključkov GPIO.	⊖ Nima razhroščevalnika, ⊖ nima vgrajenega EEPROM pomnilnika.
Arduino UNO	⊕ Zelo razširjen, velika spletna skupnost, ⊕ veliko že napisanih knjižnic, ⊕ enostaven za programiranje, ⊕ vgrajen EEPROM pomnilnik.	⊖ Majhno število priključkov GPIO, ⊖ majhna zmogljivost ⊖ samo trije časovniki.

Tabela 3.1: Prednosti in slabosti posameznih razvojnih plošč.

Raspberry Pi ima med naštetimi najzmogljivejši procesor – 900 MHz – štirijedrni ARM Cortex-A7 in največ delovnega spomina, kar 1 GB. Ima tudi 40 priključkov GPIO, nima pa razhroščevalnika.

Arduinove dobre lastnosti so enostavnost, razširjenost in že napisane knjižnice za večino dodatne strojne opreme. Žal vsebuje majhno število priključkov GPIO pa tudi zmogljivost procesorja ni tako zelo velika, saj uporablja 8-bitni 16 MHz procesor AVR ATmega328P [6].

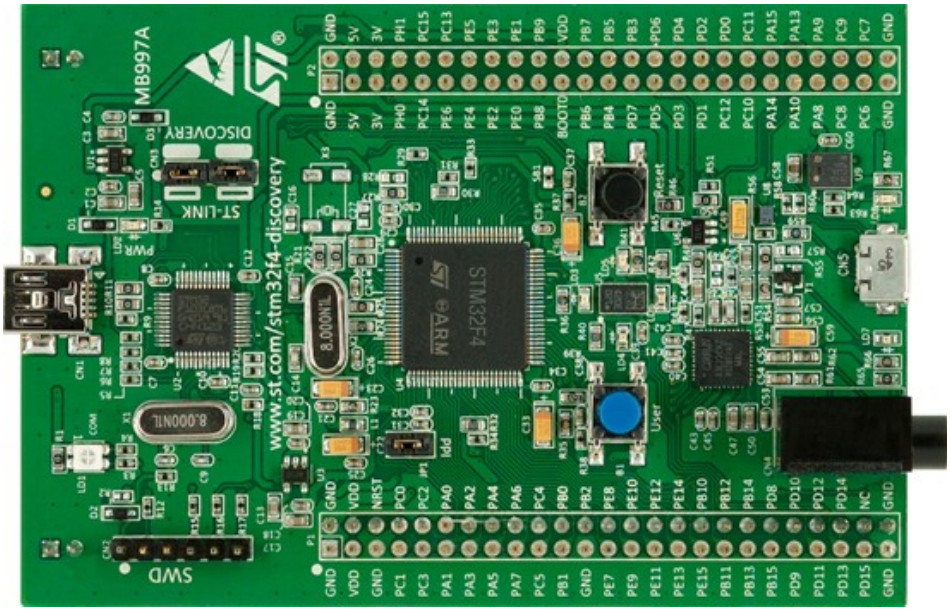
STM32F4 Discovery uporablja 32-bitni procesor Cortex M4F, ki je veliko zmogljivejši od ATmega328 v Arduinu. Ima tudi 100 priključkov GPIO, na voljo je veliko razvojnih orodij (IDE) z dobrim razhroščevalnikom. Sicer nima vgrajenega pomnilnika EEPROM, ima pa 1 MB velik bliskovni pomnilnik, ki povsem zadostuje našim potrebam. Glede na zapisane lastnosti krmilnih enot smo izbrali razvojno ploščo STM32F4 Discovery z mikrokrmilnikom STM32F407VGT6.

3.2.1 Razvojna plošča STM32F4 Discovery

Na razvojni plošči STM32F4 Discovery (Slika 3.1) je 32-bitni mikrokrmilnik STM32F407VG6, ki skupaj s centralno procesno enoto vsebuje še veliko drugih elementov oziroma naprav [7]:

- jedro: ARM® 32-bitni Cortex® M4 CPE s strojno podporo za plavajočo vejico, ura procesorja 168 MHz;
- 1 MB bliskovnega pomnilnika;
- 192 KB + 4 KB pomnilnika SRAM, 64-KB CCM podatkovni pomnilnik RAM;
- do 14 časovnikov: dvanajst 16-bitnih in dva 32-bitna časovnika do 168 MHz, vhod za inkrementalni dajalnik pozicije;
- vgrajen razhroščevalnik ST-LINK/V2-A;
- serial wire debug (SWD) in vmesnik JTAG;
- 100 GPIO priključkov s podporo prekinitvam;
- trije vmesniki I^2C;
- štirje vmesniki USART/2 UART (10,5 Mb/s);
- trije vmesniki SPI (42 Mb/s);
- dva vmesnika CAN (2,0 B);
- USB ST-LINK s tremi različnimi vmesniki:
    - razhroščevalni vmesnik,
    - navidezni serijski vmesnik,
    - podpora za podatkovni disk;
- napajanje prek vodila USB ali zunanje 5 V napajanje;
- izhodna napetost za napajanje zunanjih aplikacij: 3 V in 5 V;
- LED dioda LD1 (rdeča/zelena) za USB komunikacijo;
- LED dioda LD2 (rdeča) za 3,3 V napajanje;

- štiri uporabniške LED diode, LD3 (oranžna), LD4 (zelena), LD5 (rdeča) in LD6 (modra);
- dve tipki: uporabniška (angl. user) in tipka za ponastavitev (angl. reset).



Slika 3.1: Razvojna plošča STM32F4 Discovery [3].

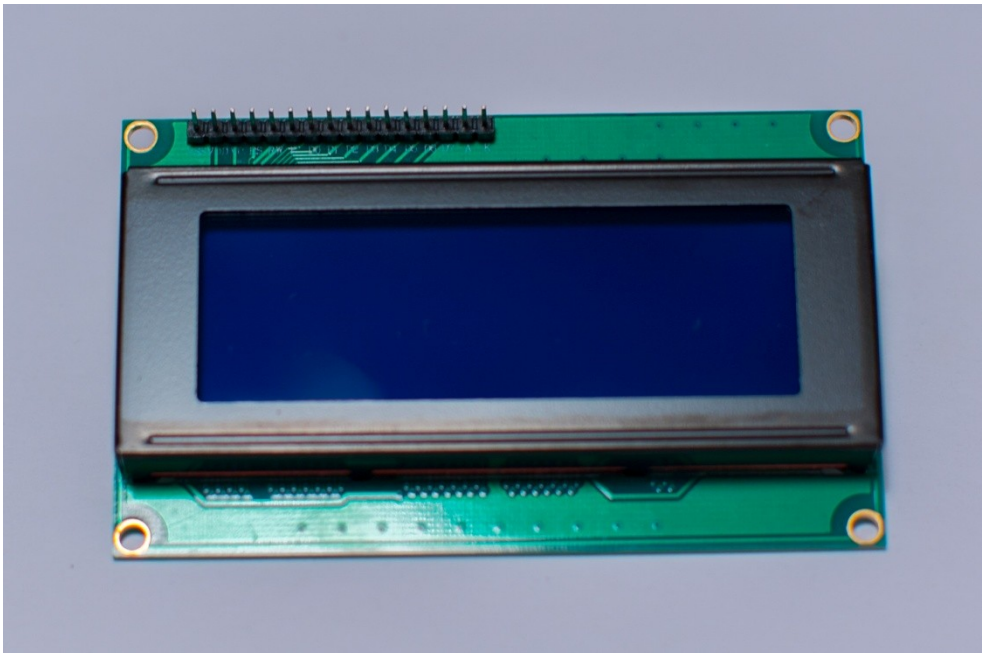
3.3 Ostala strojna oprema

Poleg krmilne enote potrebujemo še nekaj dodatne opreme: temperaturna tipala, zvočnik, vhodno izhodni močnostni elementi, tipke in zaslon LCD, ki služijo za interakcijo z uporabnikom.

3.3.1 Modul LCD 20x4

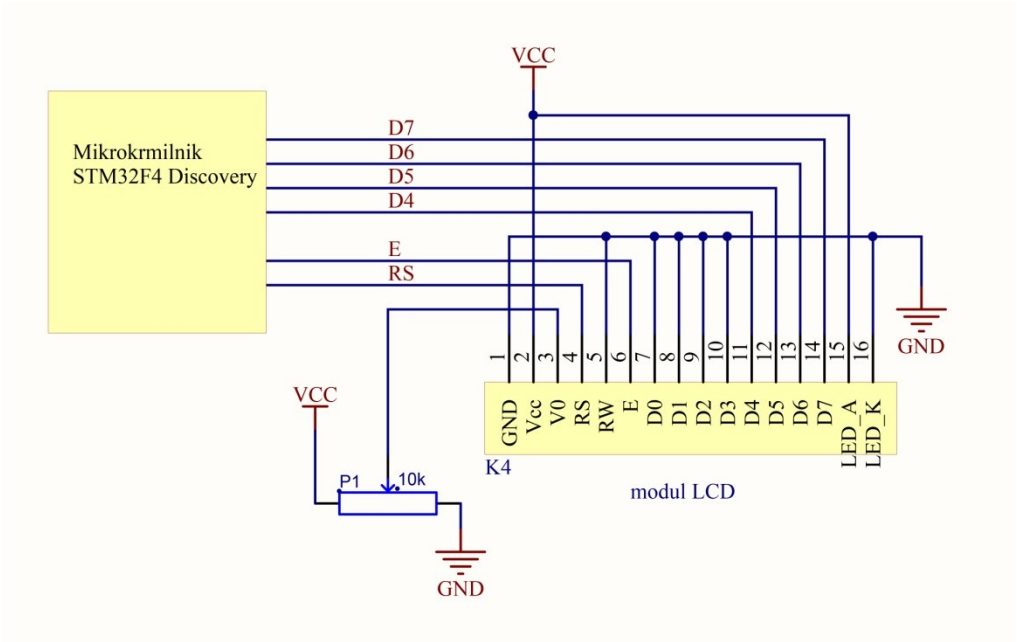
Napravo upravljamo prek tipk, na zaslonu LCD (Slika 3.2) pa se izpisujejo vse informacije kot npr. pogoji za vklop ali izklop trenutno izbrane obtočne črpalke in stanje črpalke.

Zaslon ima 4 vrstice, v vsako vrstico lahko vpišemo največ 20 znakov.



Slika 3.2: Zaslon LCD velikosti 20 × 4.

Modul ima vgrajen zelo razširjen krmilnik LCD Hitachi HD44780, ki se sicer uporablja tudi za zaslone velikosti 16 × 2. Za programiranje zaslona uporabimo knjižnico TM STM32F4 HD44780 [8] in dodamo nekaj funkcij za izpis števil v plavajoči vejici.



Slika 3.3: Električna shema povezav v 4-bitnem načinu delovanja.

Za povezavo modula LCD z mikrokrmilnikom narišemo električno shemo (Slika 3.3). Modul ima 16 priključkov (angl. pin). GND (pin 1) povežemo na GND, Vcc (pin 2) na 5 V napajanje, V0 (pin 3) služi za priklop potenciometra, s katerim nastavljamo kontrast zaslona. RS (pin 4) služi pošiljanju kontrolnih podatkov LCD-ju in ga povežemo na razvojno ploščo STM32F4 Discovery. S pinom RW (pin 5) izbiramo, ali bomo pisali ali brali z zaslona. Na zaslon bomo samo pisali, zato ga povežemo na maso. Če bi želeli na zaslonu LCD tudi prebrati izpis, bi morali na razvojno ploščo povezati še ta pin. ENABLE (pin 6) se postavi vedno, ko mikrokrmilnik zahteva prenos podatkov na zaslon LCD. Sledi priklop podatkovnih pinov D0–D7. Modul poleg 8-bitnega ponuja možnost krmiljenja tudi preko samo štirih podatkovnih linij. Ker bomo uporabili samo 4-bitni prenos podatkov, uporabimo le zgornje štiri priključke, torej D4–D7 (pini 11–14), ostale povežemo na maso. Pina 15 in 16 sta namenjena osvetlitvi LCD-ja in ju priklopimo na napajanje. Pin 15 na Vcc, pin 16 pa na GND. Modul LCD ima na vogalih tiskanine tudi 4 luknje, kar nam pride prav pri montaži v ohišje.

3.3.2 Temperaturno tipalo DS18B20

DS18B20 je digitalno temperaturno tipalo podjetja Maxim Integrated. Če bi izbrali analogno tipalo, bi morali zagotoviti natančno in stabilno napajanje, upoštevati upornost in kapacitivnost žice ter izračunati temperaturo na podlagi napetosti in trenutne upornosti. Digitalna tipala imajo to prednost, da nam ni potrebno izračunavati temperature, saj pretvorbo izvršijo sami in nam tako pošljejo samo digitalno vrednost. Izbira digitalnega tipala je zato primernejša, uporabimo pa že napisano knjižnico TM STM32F4 DS18B20 [8]. DS18B20 [9] izmeri temperaturo, jo pretvori v digitalno vrednost in pošlje preko protokola one-wire [10]. Odčitek je lahko 9-, 10-, 11-, ali 12-biten. Pri 12-bitni ločljivosti znaša čas pretvorbe 750 ms, pri 11-bitni ločljivosti pa že pol manj (Tabela 3.2).

Ločljivost v bitih	**Čas pretvorbe**
9	93,75 ms ($t_{pretv}/8$)
10	187,5 ms ($t_{pretv}/4$)
11	375 ms ($t_{pretv}/2$)
12	750 ms (t_{pretv})

Tabela 3.2: Čas pretvorbe temperature v odvisnosti od izbrane ločljivosti.

Natančnost je reda ± 0,5 °C na obsegu −10 °C do +85 °C. Maksimalen obseg merjenja je od −55 °C do +125 °C, kar za merjenje temperature vode, ki v ogrevalnem sistemu ne sme preseči temperature vrelišča, povsem zadošča.

Za naše potrebe izberemo tipalo v vodoodpornem ohišju (Slika 3.4).

Slika 3.4: Temperaturno tipalo DS18B20 v vodoodpornem ohišju [11].

3.3.3 Piezo piskač

Če krmilni program preide v varni način delovanja, je potrebno z zvočnim signalom uporabnika opozoriti na napako. Najbolj enostavna rešitev je piezo piskač (Slika 3.5), ki oddaja visoko frekvenco zvoka (od 2500 Hz do 3800 Hz). Pri 12 V napetosti in porabi toka le 20 mA glasnost presega 80 dB, zato se zvok zelo dobro sliši. Napajalna napetost je zelo prilagodljiva: od 3 V pa vse do 18 V.

Slika 3.5: Piezo piskač [12].

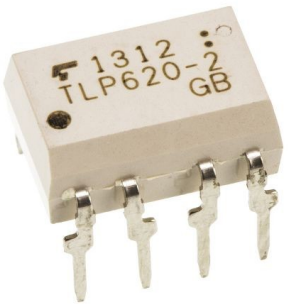
3.3.4 Izhodni močnostni elementi

Za vklop in izklop obtočne črpalke potrebujemo močnostni element, saj obtočno črpalko poganja izmenična 230V omrežna napetost. Izbirali smo med polprevodniškimi (angl. solid state relay) in klasičnimi elektromagnetnimi releji. Prednost polprevodniških relejev je v tem, da je preklop veliko hitrejši in ne povzroča iskrenja. Ne vsebujejo premikajočih se delov in imajo posledično daljšo življenjsko dobo, so tihi in odporni na vibracije. Vendar pa se polprevodniški releji grejejo. Ob upoštevanju, da bo vsa elektronika zaprta v majhno ohišje, izberemo klasični elektromagnetni rele z napetostjo tuljave 5 V. Ker želimo, da je sistem razširljiv, v elektronski sklop vgradimo 4-kanalno rele kartico (Slika 3.6) za krmiljenje morebitnih dodatnih porabnikov.

Slika 3.6: 4-kanalna rele kartica [13].

3.3.5 Vhodni močnostni elementi

Poleg 230V izhodov imamo še en 230V vhod – sobni termostat. Za zaznavanje stanja sobnega termostata, ki deluje na omrežni napetosti 230 V, izberemo AC optosklopnik z ustreznim preduporom. Optosklopnik nam zagotovi galvansko ločitev med omrežno napetostjo in mikrokrmilnikom, da ob morebitnih velikih napetostnih sunkih odpove le optosklopnik, ne pa tudi naše vezje in krmilnik. Pomembno je, da je optosklopnik namenjen priklopu na izmenični tok (AC), saj ga prek dveh uporov priklopimo neposredno na izmenično omrežno napetost. Tudi tukaj moramo predvideti morebitno kasnejše dodajanje funkcionalnosti, zato dodamo v vezje 4 optosklopnike.



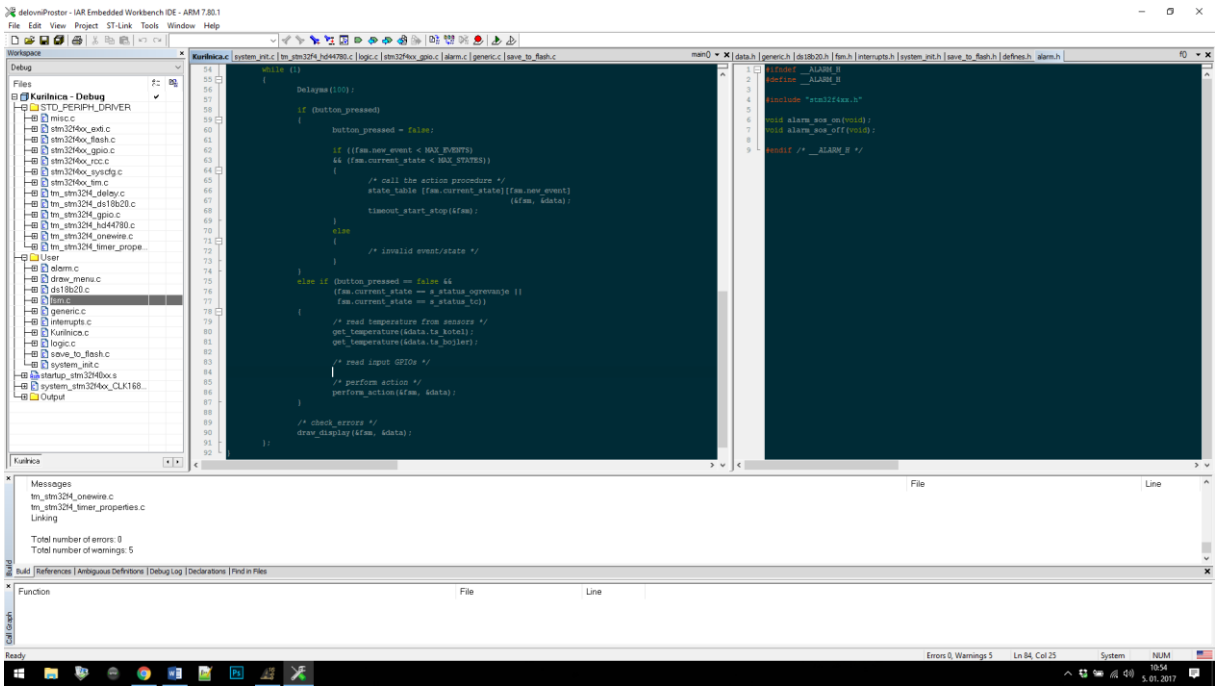
Slika 3.7: Optosklopnik TLP620-2 [14].

Izberemo optosklopnik TLP620-2 (Slika 3.7) v ohišju DIP (angl. dual in-line package) in s pripadajočim podnožjem, da ga lahko ob morebitni okvari enostavno zamenjamo.

TLP620-2 ima v ohišju že vgrajena 2 optosklopnika, zato fizično potrebujemo samo 2 taka čipa.

Poglavje 4 Razvojno okolje

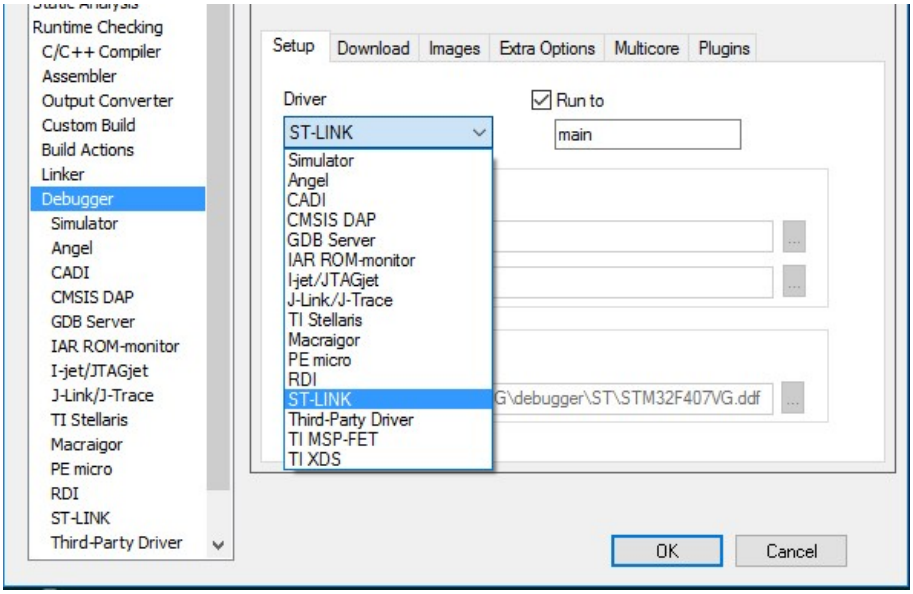
Obstaja veliko razvojnih okolij (angl. IDE) za pisanje aplikacij na procesorjih ARM Cortex M4, vendar se za to nalogo zdi najbolj primeren IAR Embedded Workbench (Slika 4.1) [15], saj v brezplačni različici ponuja polno funkcionalnost. Omejitev je le dolžina kode, ki ne sme preseči 32 kB, kar je za naše potrebe več kot dovolj.



Slika 4.1: Delovno okolje IAR Embedded Workbench.

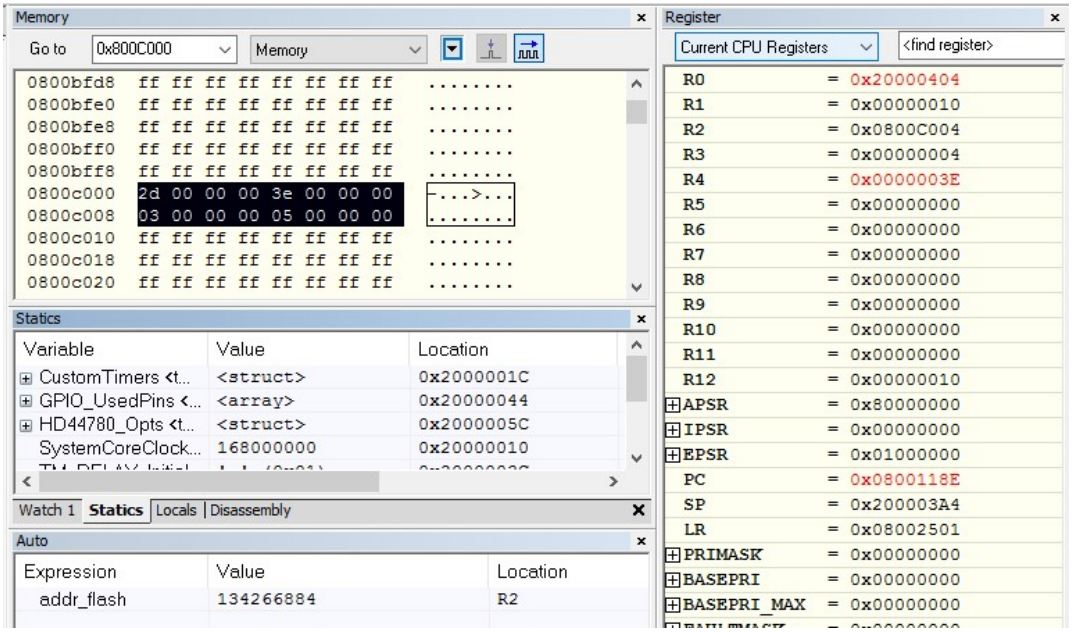
4.1 Programiranje in razhroščevanje

Za programiranje razvojne plošče STM32F4 Discovery je najprej potrebno namestiti gonilnike, ki so na voljo na uradni spletni strani podjetja STMicroelectronics. Razvojna plošča ima vgrajen ST-LINK razhroščevalni vmesnik, ki služi za programiranje in razhroščevanje (Slika 4.2). Kodo prevedemo in preko kabla USB naložimo na razvojno ploščo.



Slika 4.2: Za programiranje in razhroščevanje izberemo ST-LINK vmesnik.

Za razhroščevanje lahko uporabimo tudi simulator ST-LINK, kar nam omogoča, da nekatere programske napake odpravimo, še preden razvojno ploščo fizično priklopimo. Razhroščevalnik je zelo prilagodljiv, spremljamo lahko takorekoč katerikoli parameter in register.



Slika 4.3: Razhroščevalnik lahko prikaže vrednosti spremenljivk v bliskovnem pomnilniku .

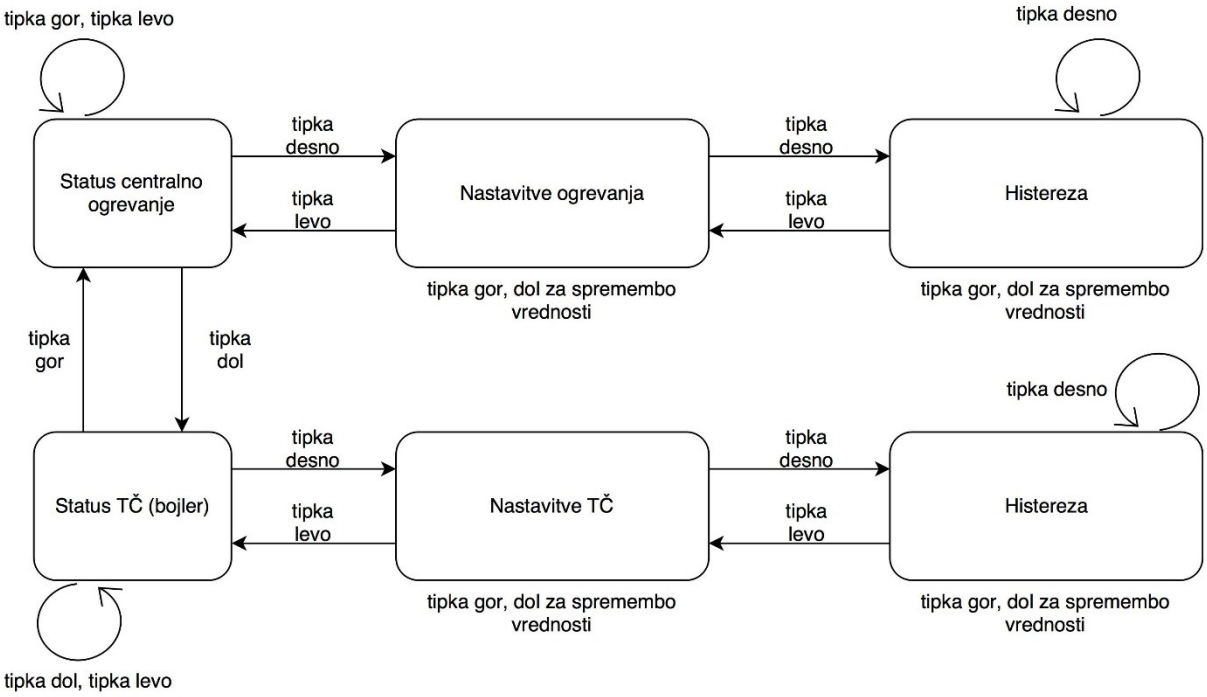
Zelo prav pride tudi neposreden vpogled v pomnilnik med samim delovanjem. Na naslovu `0x800C000` vidimo vrednosti štirih 32-bitnih spremenljivk, ki so shranjene v bliskovnem pomnilniku (Slika 4.3).

Poglavje 5 Uporabniški vmesnik

Izpis na zaslonu je edini način povratne informacije uporabniku, da ta lahko spremlja trenutne vrednosti tipal in stanja vseh izhodov. Ker uporabnik lahko spreminja več parametrov, ni mogoče prikazati vseh informacij statično na enem zaslonu. Zato razvijemo uporabniški vmesnik v obliki strukturirane množice menijev in sistemom za navigacijo med njimi. Vmesnik temelji na končnem avtomatu stanj, kjer vsako stanje predstavlja eno okno za izris na zaslonu LCD.

5.1 Zgradba uporabniškega vmesnika

Uporabniški vmesnik na zaslonu LCD je v osnovi preprost. Imamo dva statusna menija – po enega za vsako črpalko. Vsak statusni meni ima tudi svoj podmeni, v katerem nastavljamo mejne temperature in histerezo. Med različnimi meniji preklapljamo s smernimi tipkami: *gor, dol, levo, desno* (Slika 5.1).



Slika 5.1: Prehodi med posameznimi meniji s pomočjo tipk.

Če se nahajamo v statusnem meniju centralnega ogrevanja, ki prikazuje stanje obtočne črpalke centralne peči, lahko s pritiskom tipke *desno* preidemo v podmeni, kjer s pritiskom tipk *gor* ali *dol* spreminjamo vrednost parametra – minimalne mejne temperature vode v kotlu za zagon obtočne črpalke.

Če v statusnem meniju pritisnemo tipko *levo*, se ne zgodi nič, oziroma se izvede klic na funkcijo, ki ne stori ničesar.

5.2 Opis posameznih menijev

Vseh menijev je šest: dva statusna in štirje meniji za nastavitve.

Prikazujemo naslednje parametre:

- temperaturo vode T_b v bojlerju toplotne črpalke;
- temperaturo vode T_k v kotlu centralne peči;
- stanje obtočne črpalke centralnega ogrevanja *oc_kotel*;
- stanje obtočne črpalke med centralno pečjo in toplotno črpalko *oc_bojler*;
- stanje izhoda sobnega termostata.

Spreminjamo lahko štiri parametre:

- mejno temperaturo vode v kotlu T_{mk} za vklop obtočne črpalke *oc_kotel*;
- histerezo *dt_kotel* za izklop obtočne črpalke *oc_kotel*;
- mejno temperaturo vode v bojlerju toplotne črpalke T_{mb} za vklop obtočne črpalke *oc_bojler*;
- histerezo *dt_bojler* za izklop obtočne črpalke *oc_bojler*.

Če se glede na vrednost nekega parametra zahteva vklop, izpišemo besedilo *VKL*, sicer *IZK*. Nastavitveni meni je za histerezi *dt_kotel* in *dt_bojler* povsem enak, zato na koncu opišemo primer nastavljanja le za *dt_kotel*.

5.2.1 Statusni meni obtočne črpalke oc_kotel

Meni v drugi vrstici prikazuje trenutno temperaturo vode v kotlu centralne peči T_k, hkrati pa tudi, ali je glede na to temperaturo pogoj za vklop izpolnjen (Slika 5.2). Če je, se izpiše besedilo *VKL*, sicer *IZK*. Tretja vrstica prikazuje stanje sobnega termostata. V primeru da sobni termostat zahteva vklop, se izpiše *VKL*, sicer *IZK*. Zadnja vrstica prikazuje, ali je obtočna črpalka vklopljena ali izklopljena.

Slika 5.2: Statusni meni obtočne črpalke *oc_kotel*.

5.2.2 Nastavitve obtočne črpalke oc_kotel

Nastavimo lahko minimalni temperaturni prag vode v kotlu T_{mk}, pri katerem se lahko vklopi obtočna črpalka (Slika 5.3). S tem dosežemo, da po sistemu ne kroži mrzla voda. S pritiskom tipk *gor* ali *dol* nastavimo željeno temperaturo. Obseg nastavitev je omejen od +20 do +90 °C.

Slika 5.3: Nastavitveni meni obtočne črpalke *oc_kotel*.

5.2.3 Statusni meni obtočne črpalke oc_bojler

Meni prikazuje dva pogoja in izhodno stanje obtočne črpalke (Slika 5.4). V drugi vrstici se izpisuje trenutna temperatura vode v centralni peči T_k. Če je ta večja od temperature vode v toplotni črpalki T_b, potem je izpolnjen prvi pogoj, ki ga tudi izpišemo v isti vrstici.

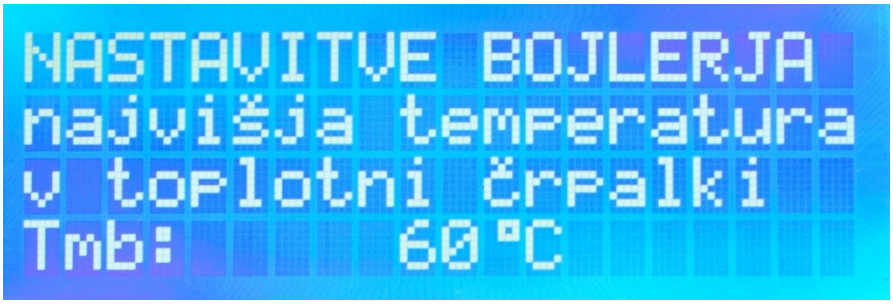
V tretji vrstici prikazujemo temperaturo vode v bojlerju toplotne črpalke, pa tudi ali je glede na to temperaturo dovoljen vklop obtočne črpalke.

V zadnji vrstici se izpisuje trenutno izhodno stanje obtočne črpalke.

Slika 5.4: Statusni meni obtočne črpalke *oc_bojler*.

5.2.4 Nastavitve obtočne črpalke oc_bojler

Nastavljamo temperaturni prag T_{mb}, pod katerim se obtočna črpalka sme vklopiti (Slika 5.5). V drugi vrstici s pritiskom na tipko *gor* ali *dol* nastavimo željeno vrednost. Ta nastavitev je potrebna, ker nočemo segrevati vode, če ima ta že primerno temperaturo.



Slika 5.5: Nastavitveni meni obtočne črpalke *oc_bojler*.

5.2.5 Nastavitev histereze dt_kotel in dt_bojler

Pogoji za vklop obtočnih črpalk so vezani na temperaturo. Ker so tipala digitalna, izmerjena vrednost temperature v kratkem časovnem intervalu lahko nekajkrat zaniha nad in pod nastavljen temperaturni prag. To povzroči večkraten hiter vklop in izklop obtočne črpalke, česar pa ne želimo. V sistem zato dodamo histerezo. Nastavljamo dve histerezi, *dt_kotel* in *dt_bojler*, vsako za svojo obtočno črpalko. Tako se lahko npr. pri temperaturi T_{mk} črpalka vklopi, pri temperaturi T_{mk} – *dt_kotel* pa izklopi. Obe histerezi uporabnik lahko nastavlja, saj neposredno vplivata na obnašanje sistema (Slika 5.6).

Slika 5.6: Primer nastavitve histereze za obtočno črpalko *oc_kotel*.

5.3 Vrnitev v statusni meni

Uporabnik lahko nastavlja različne parametre in se pozabi vrniti v statusni meni. Zato se vedno, ko uporabnik pritisne katerokoli tipko in nismo v statusnem meniju, zažene časovnik. Če se uporabnik preko tipk vrne v statusni meni, se časovnik ustavi in stanje števca se ponastavi. Če ni pritisnjena nobena tipka in se izteče vnaprej določen čas, mi pa smo še vedno v nastavitvenem meniju, se meni na zaslonu avtomatsko vrne nazaj v statusni meni. Ta funkcionalnost je pomembna, saj je program zasnovan tako, da se branje in postavljanje izhodov izvaja samo, kadar smo v statusnem meniju. Tako se ne more zgoditi, da program ne bi deloval, če bi se uporabnik pozabil vrniti nazaj.

Poglavje 6 Krmilni program

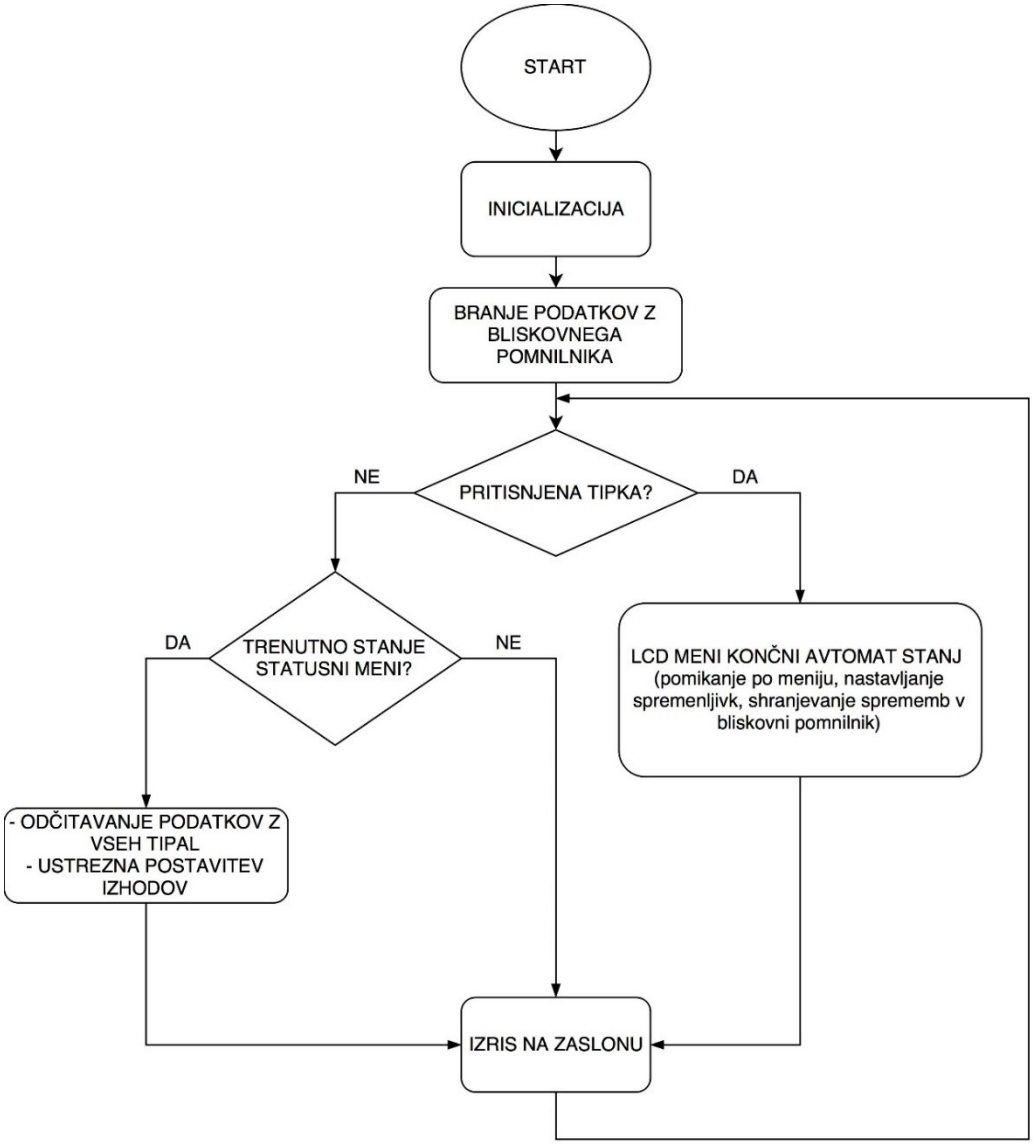
V nadaljevanju bomo najprej opisali splošen potek krmilnega programa, nato pa posamezen opis vsakega podsklopa.

Krmilni program je zadolžen za:

- inicializacijo ob vklopu,
- branje in pisanje v bliskovni pomnilnik,
- branje temperaturnih tipal,
- branje stanja sobnega termostata,
- interakcijo z uporabnikom (uporabniški vmesnik na zaslonu LCD),
- vklop ali izklop obtočnih črpalk,
- prehod v varni način delovanja v primeru napak.

Krmilni program mora biti zanesljiv, zato posebno pozornost posvetimo varnosti, saj so v primeru napake posledice lahko usodne. Upoštevati je potrebno odpoved strojne opreme in tudi smiselnost prebranih podatkov. V primeru napake zato krmilni program preide v varni način delovanja. V tem načinu se ne glede na vhodne vrednosti krmilne izhode postavi v tako stanje, da ne predstavljajo potencialne nevarnosti za celotni ogrevalni sistem.

Ko vklopimo napravo, se najprej izvedejo vse inicializacijske funkcije. Nato se preberejo vrednosti uporabniških spremenljivk, ki so shranjene v bliskovnem pomnilniku. Sledi vstop v glavno zanko programa, ki se izvede vsakih 100 ms.



Slika 6.1: Diagram poteka programa.

Kot je razvidno z diagrama (Slika 6.1), v glavni zanki najprej preverimo, če je bila pritisnjena katerakoli tipka. Če je bila, potem se izvede končni avtomat stanj za uporabniški vmesnik, ki nastavi ustrezno novo stanje avtomata.

Če tipka ni bila pritisnjena in je avtomat v stanju statusnega menija, preberemo stanje vseh vhodov in preberemo podatke z vseh tipal. Na tem mestu se glede na stanje vhodov in vrednosti tipal izvede tudi določena akcija, npr. vklop ene izmed obtočnih črpalk.

Na koncu posodobimo izpis na zaslonu LCD, nato pa se zanka spet ponovi.

6.1 Inicializacija

Ob vklopu naprave se najprej kličejo funkcije za inicializacijo (Programska koda 6.1).

```
SystemInit();
peripheral_init();
TM_DELAY_Init();
```

Programska koda 6.1: Funkcije, ki se kličejo takoj ob vklopu naprave.

`SystemInit()` konfigurira registre za sistemsko uro, vodila in nastavitve za bliskovni pomnilnik.

`peripheral_init()` funkcijo kličemo, da nastavimo priključke GPIO, inicializiramo LCD, tipke, prekinitve in časovnike. Preverimo tudi, ali so prisotna temperaturna tipala. Če niso, preidemo v varni način delovanja, kjer z zvočnim signalom in izpisom na LCD obvestimo uporabnika o napaki.

Za zakasnitve, ki se uporabijo pri programiranju zaslona LCD, uporabimo knjižnico `TM_DELAY_Init()`.

6.2 Branje iz bliskovnega pomnilnika

Takoj za inicializacijo sledi branje podatkov iz bliskovnega pomnilnika. Branje se mora vedno izvesti po vklopu naprave, saj ob izgubi napajanja spremenljivke v delovnem pomnilniku izgubijo vrednost. Tako se v slednjega naložijo tiste vrednosti, ki jih je nastavil uporabnik in so se ob nastavljanju zapisale v bliskovni pomnilnik. Za branje podatkov iz bliskovnega pomnilnika napišemo svojo funkcijo (Programska koda 6.2). Funkcija prebere podatke iz bliskovnega pomnilnika in jih zapiše v naše spremenljivke. Preberemo `u16 size` podatkov z naslova `u32 addr_flash` na naslov `struct flash_par *fpar`. Ker so podatki v strukturi definirani kot 32-bitne spremenljivke, je potrebno po vsakem branju števec `addr_flash` povečati za štiri byte.

```
int flash_read_data(struct flash_par *fpar, u16 size, u32
addr_flash)
{
      int i;
      u32 *addr_par;

      if (!fpar)
            return -1;

      addr_par = (u32 *) fpar;

      for (i = 0; i < size; i+=4)
      {
            *addr_par = *(__IO uint32_t*)addr_flash;

            addr_flash = addr_flash + 4;
            addr_par++;
      }

      return 0;
}
```

Programska koda 6.2: Funkcija za branje z bliskovnega pomnilnika.

6.3 Pisanje v bliskovni pomnilnik

Do pisanja v bliskovni pomnilnik lahko pride samo v primeru, ko uporabnik nastavi parameter in se pomakne nazaj v statusni meni. Ob vstopu v statusni meni se pred pisanjem izvede preverjanje, ali so bili podatki sploh spremenjeni. Če se podatki od trenutno nastavljenih v bliskovnem pomnilniku razlikujejo, se v tega zapiše nove vrednosti. Tako se izognemo večkratnemu pisanju; npr. če bi uporabnik nastavil neko vrednost, nato pa jo spet takoj spremenil. Pri pisanju v bliskovni pomnilnik je potrebno najprej omogočiti pisanje s funkcijo `FLASH_unlock()`, takoj za tem pa sledi brisanje celotnega sektorja (Programska koda 6.3).

```
int flash_save_data(struct flash_par *fpar, u16 size, u32
addr_flash)
{
     int i;
     u32 *addr_par;
     u8 flash_status;

     if (!fpar)
          return -1;

     flash_status = FLASH_COMPLETE;
     addr_par = (u32 *) fpar;

     FLASH_Unlock();
     FLASH_ClearFlag(FLASH_FLAG_EOP | FLASH_FLAG_OPERR
     |FLASH_FLAG_WRPERR | FLASH_FLAG_PGAERR
     |FLASH_FLAG_PGPERR | FLASH_FLAG_PGSERR);
     flash_status = FLASH_EraseSector(FLASH_Sector_3,
     VoltageRange_3);

     if (flash_status != FLASH_COMPLETE) {
          FLASH_Lock();
          return -1;
     }

     for (i = 0; i < size; i+=4)
     {
          flash_status = FLASH_ProgramWord(addr_flash,
          *addr_par);
          addr_flash = addr_flash + 4;
          addr_par++;

          if (flash_status != FLASH_COMPLETE) {
               FLASH_Lock();
               return -1;
          }
     }

     return 0;
}
```

Programska koda 6.3: Funkcija za pisanje v blok bliskovnega pomnilnika.

Nato z naslova `struct flash_par *fpar` preberemo `u16 size` vrednosti in jih zapišemo na naslov `u32 addr_flash`.

6.4 Branje tipk in generiranje dogodka

Takoj po branju spremenljivk iz bliskovnega pomnilnika vstopimo v glavno zanko programa, ki se izvrši vsakih 100 ms. V zanki se najprej preveri, ali je bila pritisnjena tipka. Preverjanje je povsem enostavno, saj so tipke vezane na zunanji prekinitveni krmilnik EXTI (ang. External interrupt). V trenutku, ko uporabnik pritisne tipko, se izvede prekinitveno servisni program (PSP) za dano prekinitev. Vsaka tipka ima svoj PSP.

```
//PSP prekinitve EXTI4: button right
void EXTI4_IRQHandler(void)
{
     if(EXTI_GetFlagStatus(EXTI_Line4)!=0)
     {
          EXTI_ClearITPendingBit(EXTI_Line4); /* clear
          interrupt flag */

          button_pressed = true;
          fsm.new_event = e_right;
     }
}
```

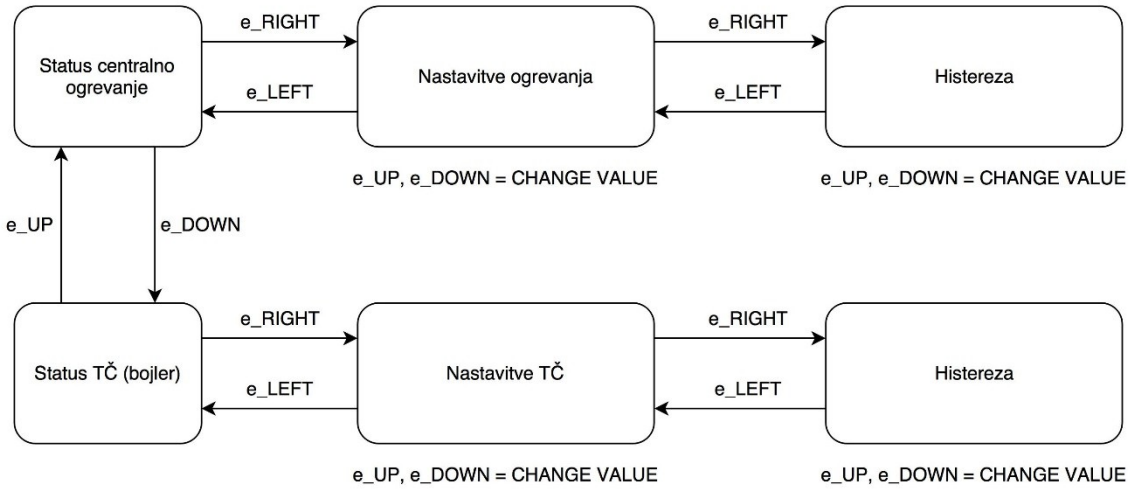
Programska koda 6.4: PSP tipke *desno*, kjer postavimo ustrezne zastavice.

Najprej pobrišemo zastavico, ki se je postavila ob prekinitvi, saj ob naslednjem prehodu skozi zanko ne želimo samodejno vstopiti v PSP. Ker mora biti koda v PSP čim krajša in enostavnejša, v tej prekinitveni rutini postavimo zastavico, s katero označimo, da je bila pritisnjena tipka. V zgornjem primeru (Programska koda 6.4) vidimo, da je bila pritisnjena tipka *desno*, zato se poleg postavljanja zastavice v strukturo *fsm* zapiše še, katera tipka je bila pritisnjena.

6.5 Končni avtomat stanj

Za premikanje oziroma navigacijo po uporabniškem vmesniku uporabimo končni avtomat stanj (Slika 6.2). Če bi kasneje želeli razširiti funkcionalnost vmesnika in bi hoteli dodati še en podmeni, to enostavno storimo tako, da v končnem avtomatu stanj definiramo novo stanje in dogodke, ki so povezani s tem stanjem. V določenem stanju smo, dokler ne pride do dogodka,

ki ga lahko spremeni. Morda samo spremenimo neko vrednost in posodobimo izpis na zaslonu, lahko pa nov dogodek pomeni prehod v drugo stanje.



Slika 6.2: Shema končnega avtomata stanj za uporabniški vmesnik.

Končni avtomat stanj v programskem jeziku C realiziramo s pomočjo dvodimenzionalne tabele kazalcev na funkcije, ki izvedejo akcije med posameznimi stanji. Kot vhod v avtomat stanj prejmemo dogodek, ki se je generiral ob pritisku na tipko. Najprej definiramo vsa stanja in dogodke, pomagamo si z naštevnim tipom spremenljivk (Programska koda 6.5).

```
enum states {s_status_kotel, s_set_kotel, s_set_d_kotel,
s_status_tc, s_set_tc, s_set_d_tc, N_STATES };

enum events { e_up, e_down, e_left, e_right, N_EVENTS };
```

Programska koda 6.5: Deklaracija stanj in dogodkov končnega avtomata stanj.

V tabeli končnega avtomata stanj (Programska koda 6.6) hranimo kazalce na funkcije, ki ne vračajo rezultata, prejmejo pa kazalec na strukturo *fsm* in *data*.

```
void (*const state_table [N_STATES][N_EVENTS]) (struct fsm
*fsm, struct data *data);
```

Programska koda 6.6: Deklaracija tabele končnega avtomata stanj.

Ko avtomat prejme nov dogodek, torej kadar se pritisne tipka in generira ustrezen dogodek, se izvede tabela končnega avtomata. Ker za stanja in dogodke uporabljamo enumeracijo, nam vsak nov dogodek v določenem stanju predstavlja klic funkcije v tabeli (Programska koda 6.7).

```
state_table [fsm.current_state][fsm.new_event] (&fsm, &data);
```

Programska koda 6.7: Klic funkcije v tabeli končnega avtomata stanj.

Novo stanje morda potrebuje naslednje podatke o končnem avtomatu: trenutno stanje avtomata, nov dogodek, podatek o pretečenem času od zadnjega dogodka in podatke o spremenljivkah, ki jih bomo nastavljali ali brali. Vsi potrebni podatki so shranjeni v dveh strukturah (Programska koda 6.8).

```
struct fsm
{
     enum states current_state;
     enum events new_event;
     enum timer timeout;
};

struct data
{
     struct flash_par fpar;
     struct temp_sensor ts_kotel;
     struct temp_sensor ts_bojler;
     u8 oc_ogrevanje;
     u8 oc_bojler;
};
```

Programska koda 6.8: Organizacija podatkov v strukturah.

Struktura *fsm* vsebuje podatke končnega avtomata stanj, *data* pa vsebuje ostale podatke o sistemu (temperaturne odčitke, mejne vrednosti in stanja obtočnih črpalk). Obe strukturi podamo kot parameter ob klicu funkcije, uporabimo pa prenos po referenci.

6.6 Vklop in izklop obtočnih črpalk

Pogoje za vklop oziroma izklop obtočnih črpalk se preverja vsakič, ko se izvrši glavna zanka programa in se nahajamo v enem izmed statusih menijev. Če pogoji za vklop ali izklop niso izpolnjeni, stanja izhoda ne spreminjamo. Ob naslednji iteraciji oziroma prehodu skozi glavno zanko se bodo pogoji morda spremenili in tako spremenili stanje.

6.6.1 Pogoji za vklop ali izklop obtočne črpalke oc_kotel

Najprej preverimo, kakšno je trenutno stanje obtočne črpalke. Če je izklopljena, preverimo, ali jo je potrebno vklopiti. Za to morata biti izpolnjena dva pogoja (6.1).

$$(sobni\_termostat == vklopljen) \wedge (T_k \geq T_{mk}) \rightarrow vklop\ oc\_kotel \tag{6.1}$$

Temperatura vode v kotlu T_k mora biti dovolj visoka, da ne kroži po sistemu mrzla voda. To preverimo s klicem funkcije `kotel_chk_tmp_on(data)`. Ta funkcija preveri, ali je izmerjena temperatura večja ali enaka tisti, ki jo je nastavil uporabnik (T_{mk}).

Drugi pogoj pa je, ali je sobni termostat javil, da je sploh potrebno ogrevati. To preverimo s klicem funkcije `thermostat_on(data)`. V tej funkciji preverimo stanje priključka GPIO, na katerega je prek optosklopnika povezan sobni termostat. Če sta torej oba pogoja izpolnjena, ustrezni priključek GPIO sproži izhodni rele in obtočna črpalka se vklopi. V strukturi podatkov na koncu še posodobimo novo stanje črpalke.

Če je obtočna črpalka že vklopljena, preverjamo pogoje za izklop (6.2). Za izklop mora biti izpolnjen vsaj en izmed dveh pogojev: sobni termostat javi izklop ali temperaturni pogoj ni izpolnjen.

$$(sobni\_termostat == izklopljen) \vee (T_k < (T_{mk} - dt\_kotel)) \rightarrow izklop\ oc\_kotel \tag{6.2}$$

Pri slednjem moramo upoštevati še nastavljeno histerezo *dT_kotel*, ki jo nastavi uporabnik.

Temperaturni pogoj za izklop obtočne črpalke centralnega ogrevanja je torej izpolnjen takrat, ko je temperatura vode v kotlu T_k za *dT_kotel* nižja od nastavljene mejne temperature T_{mk}.

6.6.2 Pogoji za vklop in izklop obtočne črpalke oc_bojler

Podobno kot prej najprej preverimo stanje obtočne črpalke. Ta podatek hranimo v spremenljivki *oc_bojler*.

Če je črpalka izklopljena, preverimo, ali jo je potrebno vklopiti (6.3). Pogoja za vklop sta tudi tu dva.

$$\big((T_k - T_b) > 0\ \big) \wedge \big(T_b < (T_{mb} - dt\_kotel)\big) \rightarrow vklop\ oc\_bojler \tag{6.3}$$

Prvi pogoj je, da je temperatura vode v kotlu (T_k) vedno večja od temperature vode v bojlerju (T_b) toplotne črpalke. Tako zagotovimo, da s toplotno črpalko ne bomo nikoli ogrevali vode v centralni peči, ampak samo vodo v bojlerju toplotne črpalke.

Drugi pogoj je, da mora biti trenutna temperatura vode v bojlerju toplotne črpalke T_b pod razliko mejne temperaturo T_{mb} in histereze *dt_bojler*. Oba pogoja morata biti izpolnjena, da se izvrši vklop obtočne črpalke.

Če je obtočna črpalka *oc_bojler* že vklopljena, preverimo pogoje za izklop (6.4).

$$\big((T_k - T_b) \leq 0\ \big) \vee (T_b \geq T_{mb}) \rightarrow izklop\ oc\_bojler \tag{6.4}$$

Pogoj za izklop obtočne črpalke med centralno pečjo in toplotno črpalko je torej izpolnjen, ko je temperaturna razlika vode med obema manjša ali enaka 0, ali pa je temperatura vode v toplotni črpalki večja ali enaka nastavljeni maksimalni temperaturi vode v bojlerju T_{mb}.

6.7 Risanje menija

Ker imamo vse podatke shranjene v ustreznih strukturah, je risanje menija trivialno. Vedeti moramo, v katerem stanju se trenutno nahajamo – ta podatek je v strukturi končnega avtomata stanj. Ko preverimo stanje, enostavno kličemo funkcijo za izris (Programska koda 6.9). Seveda potrebujemo tudi podatke za izpis, zato jih funkciji ob klicu posredujemo.

```
void draw_display(struct fsm *fsm, struct data *data)
{
     switch(fsm->current_state)
     {
     case s_status_kotel:     draw_status_kotel(data);
     break;

     case s_set_kotel:        draw_set_kotel(data);
     break;

     case s_set_d_kotel:      draw_set_d_kotel(data);
     break;

     case s_status_bojler:    draw_status_bojler(data);
     break;

     case s_set_bojler:       draw_set_bojler(data);
     break;

     case s_set_d_bojler:     draw_set_d_bojler(data);
     break;

     default:                 draw_status_kotel(data);
     }
}
```

Programska koda 6.9: Funkcija za izris na zaslon LCD.

Sicer v ogrevalnem sistemu ne pričakujemo negativnih temperaturnih vrednosti, pa vendar je potrebno poskrbeti, da se program ne ustavi, če pride do tega, saj želimo robusten, univerzalen program, ki lahko deluje kjerkoli. Spremenljivke, ki hranijo temperaturne vrednosti, so zapisane v plavajoči vejici (angl. float). Ker tak podatkovni tip ni primeren za izpis na zaslonu z 20 znaki v širino, se s pomočjo *math.h* knjižnice in funkcije *modf* to spremenljivko razdeli na dve predznačeni številki celoštevilskega podatkovnega tipa (angl. signed integer*)* – eno za celoštevilski del rezultata, drugo za decimalni del (Programska koda 6.10).

```
void float_to_two_integers(float number, s8 *int_part, s8
*dec_part)
{
     double intpart;
     double decpart;

     decpart = modf(number, &intpart);

     *int_part = (s8) intpart;
     *dec_part = (s8) (decpart * N_DECIMAL_POINTS_PRECISION);
}
```

Programska koda 6.10: Funkcija za pretvorbo tipa *float* v dve spremenljivki tipa *signed integer*.

Čeprav decimalnega dela ne potrebujemo, ga vseeno izpisujemo, da uporabnik lahko spremlja hitrost naraščanja ali padanja temperature. Ker je natančnost temperaturnega tipala v obsegu ± 0,5 °C, je povsem dovolj, da decimalni del zaokrožimo na eno decimalno mesto. Na zaslonu tako izpišemo predznak, potem celoštevilski del in nato decimalni del.

6.8 Ravnanje v primeru napak

Programsko opremo je potrebno napisati robustno, da se v primeru napake program ne zaustavi ali preide v nepredviden način. To je potrebno predvsem pri tistih aplikacijah, ki krmilijo strojno opremo, saj lahko pride do različnih nesreč, npr. požar, izpust strupenih plinov, uničenje drage opreme ... Zato je nujno, da se predvidi napake na tipalih, napake aktuatorjev, uporabniške napake ipd.

Obnašanje programa v primeru napake je odvisno od napake same, zato za vsak tip napake napišemo svojo funkcijo, ki se kliče in izvede ustrezne akcije. Če napaka ni kritična, uporabnika opozorimo nanjo z izpisom na zaslonu, program pa lahko teče naprej. V primeru kritične napake, kot je npr. odpoved temperaturnega tipala, pa program preide v varni način, kjer vse izhode postavimo na take vrednosti, da ne predstavljajo potencialne nevarnosti za sistem.

6.8.1 Preverjanje temperaturnih tipal

V našem primeru se preverja ali so temperaturna tipala prisotna in ali dajejo smiselne vrednosti. Vedno, ko želimo izmeriti temperaturo, začnemo meriti čas. Če v določenem času ne dobimo rezultata, pomeni, da je s tipalom nekaj narobe. Da preverimo smiselnost odčitanih temperaturnih vrednosti, vnaprej določimo temperaturni obseg. Če so temperaturne vrednosti zelo velike ali zelo majhne, torej izven določenega temperaturnega obsega, to pomeni, da je na tipalu napaka. V tem primeru ni mogoče nadaljevati s programom, zato preidemo v varni način delovanja.

6.8.2 Preverjanje bliskovnega pomnilnika

Preverimo tudi vrednost spremenljivk, ki jih preberemo iz bliskovnega pomnilnika, saj obstaja možnost, da se ta zaradi omejenega števila bralno-pisalnih ciklov izrabi in tako ob vklopu naprave preberemo povsem napačne vrednosti spremenljivk, bodisi prevelike bodisi premajhne. Funkcija `flash_chk_data()` ob vsakem branju preveri, ali so prebrane vrednosti v normalnih okvirih. Če niso, preidemo v varni način delovanja.

6.8.3 Varni način delovanja

Kadar sistem ne dobi dovolj podatkov za izračun izhodnih vrednosti (napaka v programu, odpoved temperaturnih tipal), ali pa so podatki nesmiselni, preidemo v varni način delovanja. Kliče se funkcija `action_safe_state(struct fsm *fsm, struct data *data)`, kjer vklopimo zvočni alarm, na zaslon izpišemo napako in ustrezno nastavimo vrednosti izhodov (Programska koda 6.11). Varni način delovanja smatra vse podatke za neveljavne, zato zaradi varnostnih razlogov vklopimo obtočno črpalko centralnega ogrevanja *oc_kotel*. Če bi namreč temperatura vode v kotlu naraščala, črpalko pa bi izklopili, bi voda lahko zavrela in tlak bi naraščal. Obtočno črpalko oc_*bojler* v varnem načinu izklopimo, saj ne predstavlja nevarnosti za sistem.

```
void action_safe_state(struct fsm *fsm, struct data *data)
{
     /* enable sound alarm */
     alarm_on();

     /* turn OFF oc_bojler (TČ) */
     GPIO_SetBits(OC_BOJLER_PORT, OC_BOJLER_PIN);
     data->oc_bojler = OFF;

     /* turn ON oc_kotel */
     GPIO_ResetBits(OC_KOTEL_PORT, OC_KOTEL_PIN);
     data->oc_kotel = ON;
}
```

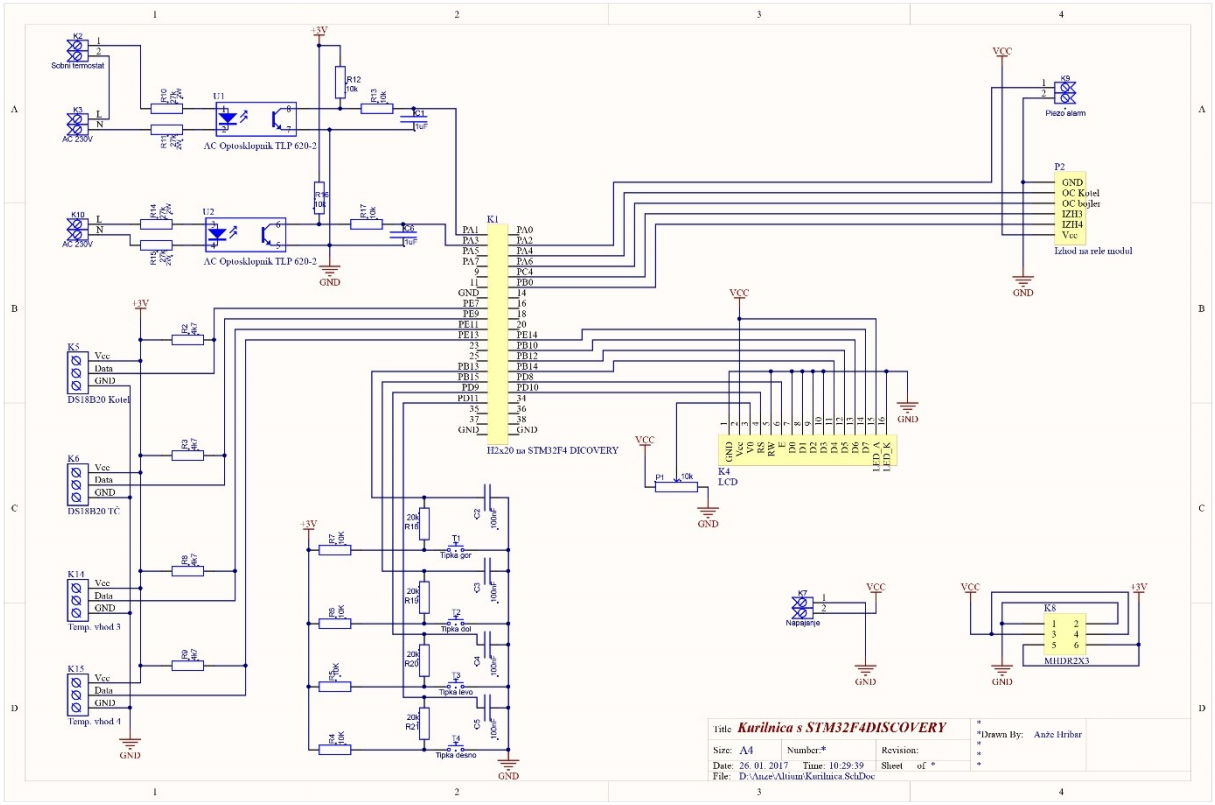
Programska koda 6.11: Funkcija za prehod v varni način delovanja.

Poglavje 7 Sestava in vgradnja prototipa

Na koncu moramo narediti tiskano vezje, vse komponente vgraditi v ohišje in temeljito testirati. Pri načrtovanju vezja moramo biti še posebno pozorni. Vezje moramo načrtovati tako, da je čim bolj odporno proti motnjam iz okolice. Vklop gorilnika za olje, obtočnih črpalk, polža za pelete in drugih naprav lahko generira električne konice ali elektromagnetne motnje, ki povzročijo nedelovanje sistema. Vsak sklop mora zato imeti na napajalnih linijah blokirne kondenzatorje, povezave med sklopi morajo biti čim krajše, povezovalni kabli pa morajo biti oklopljeni. Krmilni del mora biti fizično čim bolj oddaljen od močnostnega dela in elementov, ki so priklopljeni na omrežno napetost.

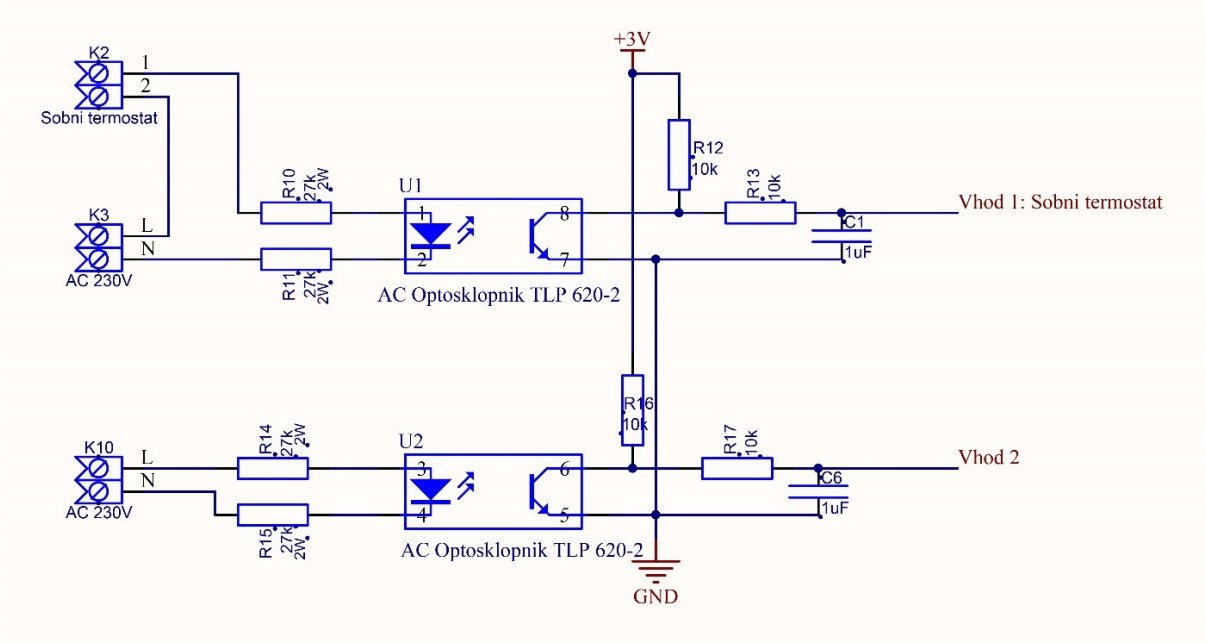
7.1 Električna shema in povezovanje komponent

Ker razvojna plošča STM32F4 Discovery ni primerna za neposreden priklop vseh vhodnih in izhodnih elementov, je potrebno načrtovati tiskano vezje (Slika 7.1).



Slika 7.1: Električna vezalna shema.

Na vhod optosklopnika pripeljemo omrežno napetost 230 V, ki je sinusne oblike. Z uporabo AC optosklopnika zagotovimo, da fototranzistor v optosklopniku prevaja obe polperiodi. Na izhodu optosklopnika napetost torej ni enosmerna temveč pulzirajoča, kot posledica izmenične omrežne napetosti na vhodu. Da to napetost zgladimo, na izhod optosklopnika zaporedno priključimo kondenzator in upor (Slika 7.2).



Slika 7.2: Vezje za zaznavanje stanja sobnega termostata.

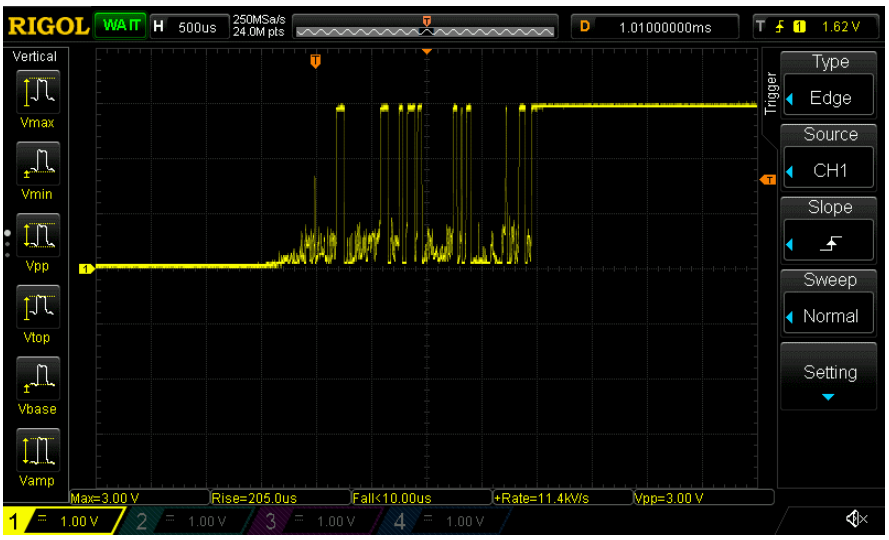
Ker optosklopnik ni namenjen neposrednemu priklopu na omrežno napetost U, potrebujemo predupor, ki bo to napetost zmanjšal. Vrednost predupora R_p izračunamo po Ohmovem zakonu (7.1). Iz podatkovnega lista optosklopnika je razvidno, da mora imeti tok I_{LED}, ki teče skozi LED diodo v optosklopniku vrednost 4 mA. Pri tem toku je padec napetosti U_{LED} na LED diodi enaka 1,1 V, torej mora biti vsa ostala napetost na preduporu.

$$R_p = \frac{U - U_{LED}}{I_{LED}} = \frac{230\ \text{V} - 1{,}1\ \text{V}}{0{,}004\ \text{A}} = 57225\ \Omega \tag{7.1}$$

Predupor mora torej imeti vrednost približno 60 kΩ. Moč, ki se troši na preduporu, je približno 1 W. Da porazdelimo disipacijo, zaporedno povežemo dva upora moči 2 W in vrednosti 30 kΩ. S tem razbremenimo predupor, saj se na enem 2W preduporu troši moč 0,5 W.

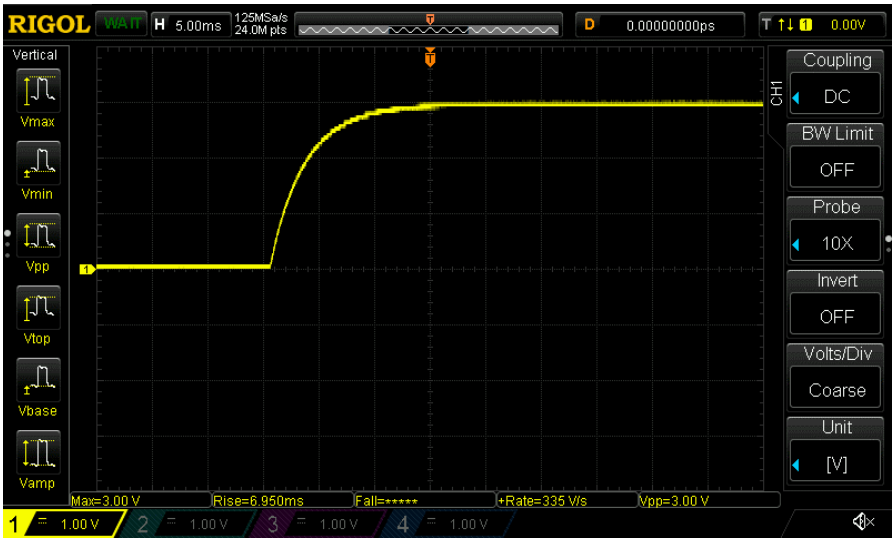
Temperaturni tipali DS18B20 sta priključeni vsak na svoj priključek GPIO, prek dvigovalnega (angl. pull-up) upora. Za možnost kasnejše razširitve dodamo še dve taki vezji. Temperaturni tipali do ohišja napeljemo s kabli UTP.

Tipke priklopimo prek dvigovalnih uporov vrednosti 10 kΩ. Ob pritisku na tipko napetost večkrat zaniha, kar predstavlja težavo, saj zunanji prekinitveni krmilnik to zazna kot več pritiskov (Slika 7.3).



Slika 7.3: Brez gladilnega vezja ob pritisku tipke napetost večkrat zaniha.

Za odpravo težave vzporedno na stikalo priklopimo kondenzator vrednoti 100 nF. Tako se ob pritisku tipke kondenzator izprazni na napetost 0 V, ob spustu tipke pa napetost počasi naraste na 3 V (Slika 7.4).

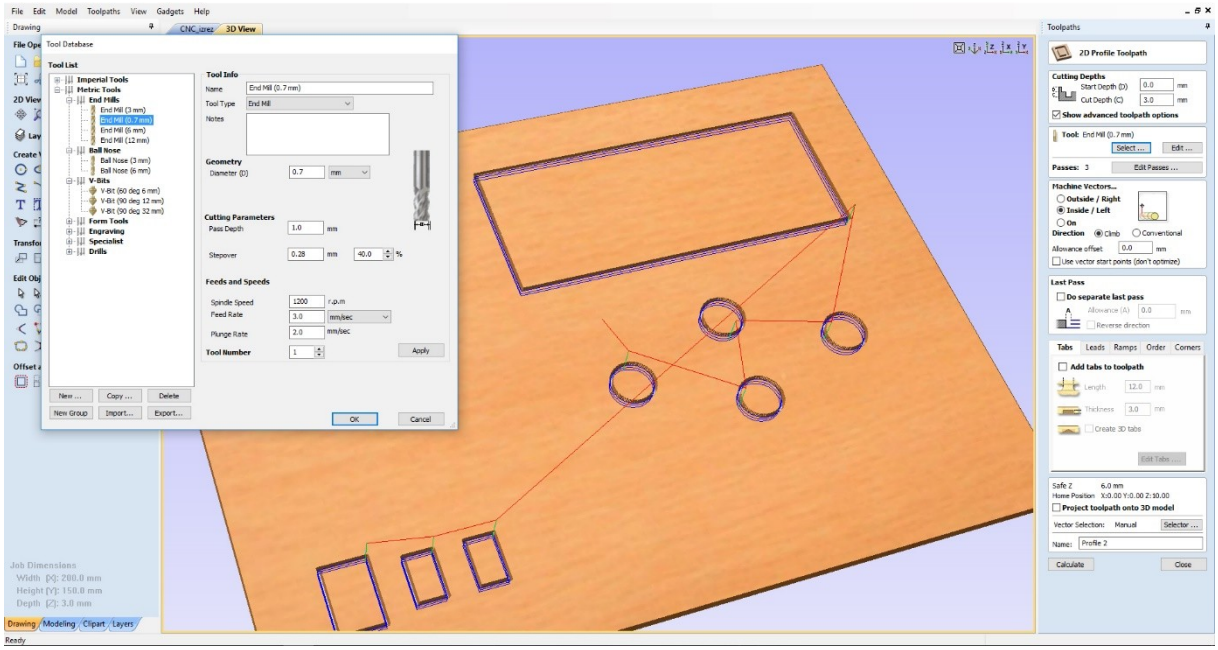


Slika 7.4: Počasno naraščanje napetosti kot posledica polnjenja kondenzatorja.

Za napajanje uporabimo 5V stikalni napajalnik, ki nam mora zagotoviti 1 A toka. LCD modul priklopimo prek oklopljenega kabla s 16 priključki. Ker se zaslonu lahko nastavlja kontrast, namesto upora uporabimo 10kΩ trimer potenciometer, s katerim lahko kasneje poljubno spremenimo kontrast.

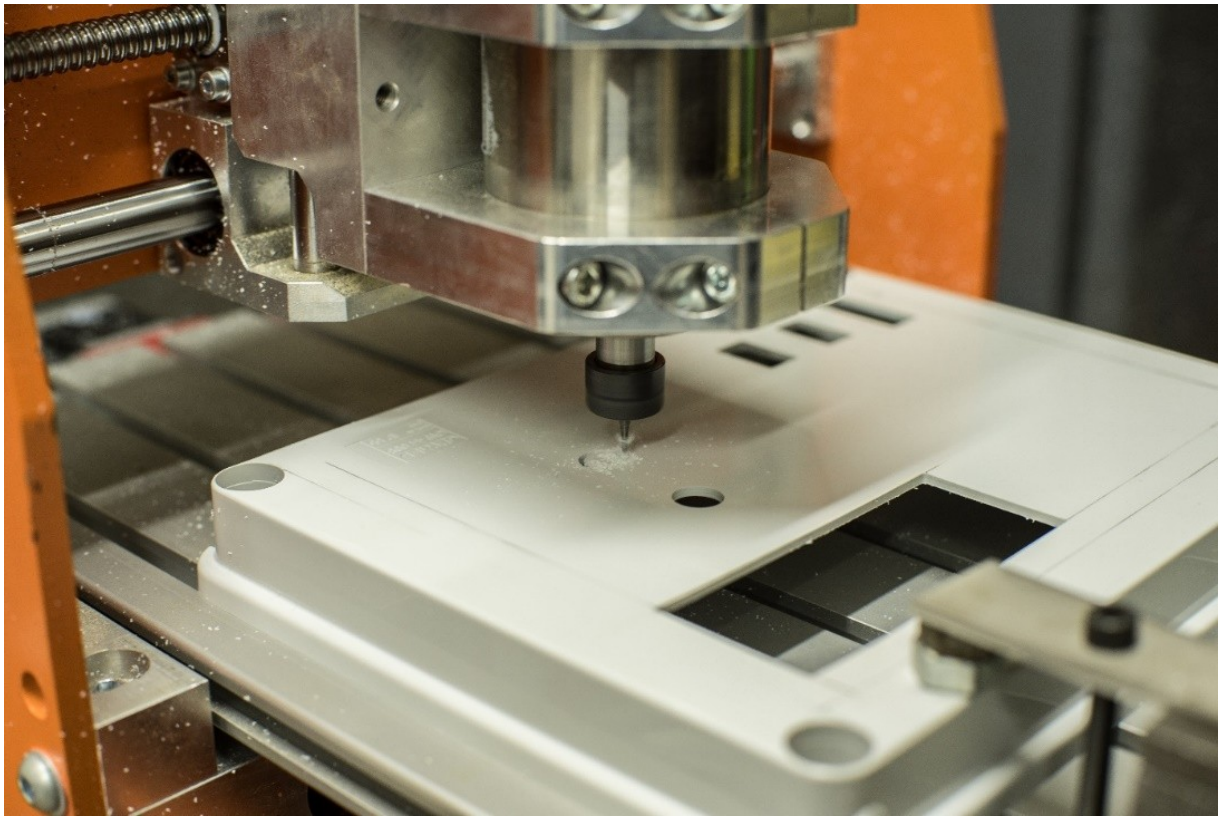
7.2 Izdelava ohišja

Za vgradnjo elektronike izberemo plastično ohišje, v katerega je potrebno izrezati pravokotne odprtine, zato uporabimo rezkalni stroj CNC. Potrebne so odprtine za zaslon LCD, kvadratni tipki za ročni vklop in izklop obeh obtočnih črpalk, štiri uporabniške tipke in glavno stikalo. V programu CAD/CAM narišemo izreze in določimo globino reza (Slika 7.5).



Slika 7.5: Predogled vseh izrezov in poti orodja v programu CAD/CAM.

Nato izberemo in določimo pot orodja ter izvozimo generirano G-kodo, ki jo poženemo na rezkalnem stroju CNC (Slika 7.6).



Slika 7.6: CNC rezkalni stroj v masko ohišja izrezuje vse potrebne odprtine.

Da se plastično ohišje med rezkanjem ne topi, uporabimo nizke obrate (okoli 1200 obratov na minuto) in majhno hitrost pomika (3 mm/s).

7.3 Testiranje

Sledi vgradnja komponent v ohišje. Ko je prototip naprave sestavljen (Slika 7.7), pred prvim priklopom še enkrat preverimo, da je vse prav povezano, nato pa testiramo osnovno delovanje naprave. Ko vse deluje, testiramo še robne pogoje, kot so npr. prevelike ali negativne temperaturne vrednosti, nedelovanje (izklop) temperaturnih tipal, stanje spremenljivk ob izklopu in takojšnjem vklopu napajanja in prehod v varni način delovanja. Ko vse deluje, prototip vgradimo v ogrevalni sistem (Slika 7.8) in preizkusimo delovanje.

Slika 7.7: Sestavljen prototip.

Slika 7.8: Prototip nameščen v ogrevalni sistem.

Poglavje 8 Sklepne ugotovitve

V tej diplomski nalogi smo rešili problem krmiljenja obtočnih črpalk v starejših ogrevalnih sistemih, katerim se doda toplotna črpalka. Lastnik delno prenovljenega ogrevalnega sistema je tako prihranil nekaj denarja, saj je naša rešitev občutno cenejša od že izdelanih, a dragih rešitev, ki so sicer na trgu.

Ker smo napravo zasnovali z mislijo dodajanja novih tipal in krmiljenja še kakšnih drugih elementov, ne le obtočnih črpalk, je mogoče zgolj s spremembo programske opreme to enostavno uresničiti.

V praksi naša rešitev deluje izvrstno. Še posebej dobra lastnost je ta, da je mogoče parametre nastavljati in si jih naprava zapomni tudi ob izgubi napajanja. Optimalne nastavitve je namreč nemogoče predvideti vnaprej. Lastnik je poleg programskega pogrešal tudi možnost ročnega vklopa in izklopa obtočnih črpalk, zato smo kasneje dodali tudi to možnost.

Pri prvi vgradnji je nastala težava, saj smo za zaznavanje stanja sobnega termostata sprva uporabili elektromagnetni rele. Ob preklopu je ta generiral toliko elektromagnetnih motenj, da so se na zaslonu LCD izpisovali naključni znaki, vendar smo težavo uspešno rešili z uporabo optosklopnika in oklopljenih povezovalnih kablov.

Prototip naprave je zanesljivo deloval skozi celotno ogrevalno sezono. V nadaljevanju nameravamo poleg obtočnih črpalk krmiliti še peletni gorilnik.

Naprava ima veliko možnosti za nadaljnji razvoj. Z ustreznim modulom v primeru napake ali spremembe parametrov povsem enostavno dodamo možnost obveščanja prek storitve kratkih sporočil (angl. SMS), doda se lahko tudi možnost izpisa vseh podatkov na lokalnem strežniku prek protokola TCP/IP, krmiljenje naprave s pametnim telefonom in zajem podatkov na zunanjo pomnilniško kartico.

Literatura

[1] B. Grobovšek. (2010). Legionele - Ukrepi za njihovo preprečevanje pri pripravi tople sanitarne vode. *ENSVET Energetsko svetovanje* [Online]. Dosegljivo: http://gcs.gi-zrmk.si/Svetovanje/Clanki/Grobovsek/PT98.htm. [Dostopano: 5. 1. 2017]

[2] B. Breščak et al. Razhroščevalniki. *Projekt e-računalništvo* [Online]. Dosegljivo: http://www.s-sers.mb.edus.si/gradiva/rac/moduli/programirljive_naprave/04_napake/05_datoteka.html. [Dostopano: 15. 12. 2016]

[3] STMicroelectronics. *STM32F4DISCOVERY* [Online]. Dosegljivo: http://www.st.com/en/evaluation-tools/stm32f4discovery.html. [Dostopano: 12. 12. 2016]

[4] R. P. Foundation. *Raspberry Pi 2 Model B* [Online]. Dosegljivo: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/. [Dostopano: 12. 12. 2016]

[5] *Arduino, Arduino & Genuino Uno* [Online]. Dosegljivo: https://www.arduino.cc/en/Main/ArduinoBoardUno. [Dostopano: 12. 12. 2016]

[6] *Atmel, ATmega328* [Online]. Dosegljivo: http://www.atmel.com/devices/ATMEGA328.aspx. [Dostopano: 12. 12. 2016]

[7] *STMicroelectronics* [Online]. Dosegljivo: http://www.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32f4-series/stm32f407-417/stm32f407vg.html. [Dostopano: 13. 12. 2016]

[8] T. Majerle. *All STM32F4 libraries* [Online]. Dosegljivo: https://stm32f4-discovery.net/2014/05/all-stm32f429-libraries-at-one-place/. [Dostopano: 20. 11. 2016]

[9] Maxim Integrated. *Programmable Resolution 1-Wire Digital Thermometer* [Online]. Dosegljivo: https://www.maximintegrated.com/en/products/analog/sensors-and-sensor-interface/DS18B20.html. [Dostopano: 8. 11. 2016]

[10] Maxim Integrated. *1-WIRE* [Online]. Dosegljivo: https://www.maximintegrated.com/en/products/digital/one-wire.html. [Dostopano: 16. 12. 2016]

[11] L. Hans. *Tweaking4All.* [Online]. Dosegljivo: https://www.tweaking4all.com/hardware/arduino/arduino-ds18b20-temperature-sensor/ [Dostopano: 5. 3. 2017]

[12] Piezo piskač [Online]. Dosegljivo: http://www.ic-elect.si/piezo-piskac-sep2280-3-18v-88db-h-l.html [Dostopano: 5. 3. 2017]

[13] Sain Smart. *Sainsmart 4-Channel 5V Relay Module* [Online]. Dosegljivo: https://www.sainsmart.com/4-channel-5v-relay-module-for-pic-arm-avr-dsp-arduino-msp430-ttl-logic.html [Dostopano: 5. 3. 2017]

[14] RS Components Ltd. *Toshiba TLP620-2(F) AC Input Transistor Output Dual Optocoupler* [Online]. Dosegljivo: http://uk.rs-online.com/web/p/optocouplers/6258299/ [Dostopano: 5. 3. 2017]

[15] IAR Systems. *IAR Embedded Workbench* [Online]. Dosegljivo: https://www.iar.com/iar-embedded-workbench/#!?currentTab=free-trials. [Dostopano: 10. 12. 2016]